IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 6, Ver. VIII (Nov – Dec. 2014), PP 32-38
www.iosrjournals.org

Steps to System Reconfiguration in a Distributed Database System: A Fault-tolerant Approach

Silas Abasiama Ita[1], Asagba Prince O[2]. and Igiri Chinwe Peace[3]
[1, 2, 3]*Department of Computer Science, University of Port Harcourt, Choba, Port Harcourt, Nigeria.*

Abstract: *In this paper, we examine steps to system reconfiguration in a Distributed Database using some fault tolerance scheme for a distributed database system that is compatible with respect to system failure i.e. Normal transaction processing is carried out assuming that no failure will occur, but if failure is suspected at some stage or site then a system reconfiguration is undertaken. The reconfiguration and system transaction are guaranteed as long as there exist a majority of working sites belonging to the same database*
Keywords: *Distributed Database, Reconfiguration, Homogeneous database, Database Replication, Failures*

I. Introduction

A fault is a defect in the component of a system, [2] faults results in errors, which are undesired or invalid component states. Errors may result in failures, which means loss of service expected from the component (or the system in which the component is a part). After the failure of a component (or subsystem), the system may take a corrective action called failure recovery. This may involve reconfiguring the system to isolate the failed component and to reorganize the system so that it can be restarted without the failed data; this can be achieved using some fault-tolerant techniques. Once the failed site is repaired, it is reintegrated with the system. The Fault-tolerant techniques should protect the system against any type of failure.

Failures during a transaction (whether network partitions or site failures) can be tolerated as long as [3]the sites available at commit contain a majority of replicas of all the objects written to and during reads, a majority of replicas are read to find the version numbers. If these requirements are violated, the transaction must be aborted.

A fault-tolerant control policy essentially determines whether to occupy one of the two working servers to restore the data in the failed server or to overhaul the entire system at the state of one server failure [10]. It is determined by how the designer penalizes a control action at any given state.

A distributed database system, like any other database systems are subject to failures. Despite this, any organization that depends upon a database must have that database available when it is required. Therefore, any database management system intended for a serious business or organizational used, must have adequate facilities for fast reconfiguration after failure [5]. In particular, whenever a transaction is submitted in a distributed database (i.e. a database that stores logically related database over two or more physically independent sites), the system must ensure that either:

i. All the operations in the transaction are completed successfully and their effect is recorded permanently in the database or
ii. The transaction has no effect whatsoever on the database or on any other site.

The understanding of the various fault-tolerant methods available to reconfigure after such failure is very essential to any serious study of distributed database system [5].

Therefore the objective of this paper is to provide a way to reconfigure a distributed database when there is a failure in one or more participating site(s) using some fault tolerance schemes.

The rest of this paper is structured as follows. In section II, we review some related works, Section III describes the Distributed Database system and types of failures in a DDB, while section IV discusses the Reconfiguration Approaches. In section V, we conclude this paper.

II. Related Work

Some researchers have used reconfiguration in a variety of context.

In [15], a new level wise approach was used for system reconfiguration and service restoration, where they classified fault as feeder, network line, single or multiple faults depending on the number of faults. Iteration method is used to determine an optimal compensation solution which also enable variation of the solution when network configuration changes.

[5], Used an optimistic resiliency control scheme for system reconfiguration. This scheme is suitable when subsystem failure occur rarely and for autonomous distributed and multi-database system where sites may communicate with each other at their will.

In [10], partitioned architecture of the system and data redundancy was used to restore lost data set in a single server. Control policies are determined by solving Markov decision problem with cost criteria that penalized system unavailability and slow query response.

Furthermore, [6] worked on the power of reconfiguration in fault tolerance distributed databases where they talked on the use of Dynamic reconfiguration to significantly increase fault tolerance in replicated databases.

This work differs from previous works on system reconfiguration because it presents a step by step approach to reconfiguration using some fault-tolerant schemes.

III. Distributed Database System

This is single logical database that is spread physically across computers in multiple location that are connected by a data communication network stores a logical related database over two or more physically independent sites, the sites are connected through a computer network. In a distributed database system, a database is composed of several parts which are referred to as database fragments, these fragments are located at different sites.

Distributed DBMS can also be integrated as a multiple process, single data called MPSD, to allow more than one computer to access a single database. Large corporations may require an enterprise database to support many users over multiple departments. This would require the implementation of a multiple process, multiple data scenario, or MPMD, in which many computers are linked to a fully distributed client/server DDBMS [12]. The DDBMS offer more reliability by decreasing the risk of a single site failure. If one computer in the new site fails, the workload is distributed to the rest of the computers. Furthermore, a DDBMS allows replication of data among multiple sites, data from the failed site may still be available at other sites. In a centralized database, data can be implemented as a single process, single data scenario or SPSD, in which one computer is linked to the host DBMS to retrieve data .A centralized DBMS is different because, a failed computer that houses the database will debilitate the entire system. A distributed database is a database in which storage devices are not all attached to a common processing unit such as the CPU [9], controlled by a distributed database management system(sometimes called a distributed database system). It may be stored in multiple computers, located in the same physical location; or may be dispersed over an interconnected computer. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely-coupled sites that share no physical components. System administrators can distribute collections of data (e.g. in a database) across multiple physical locations. A distributed database [13] can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. Because they store data across multiple computers, distributed databases can improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to one. A block diagram of the architecture of DBMS is given in figure 1 below.

Two processes ensure that the distributed databases remain up-to-date and current: replication and duplication.

i) **Replication:** This involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. [12]The replication process can be complex and time consuming depending on the size and number of the distributed databases. This process can also require a lot of time and computer resources.

ii) **Duplication:** This on the other hand, [12] has less complicity. It basically identifies one database as a master and duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location (sites) has the same data. In the duplication process, user may change only the master database. This ensures that local date will not be overwritten.

Both replication and duplication can keep the data current in distributive locations. Besides distributed database replication and fragment, there are many other distributed database reconfiguration design technologies. For example, local autonomy, synchronous and asynchronous distributed database technologies. These technology implementations depend on the needs of the business and the sensitivity/confidentiality of the data stored in the database, and hence the price the business is willing to spend on ensuring data security, consistency and integrity [13]. Distributed DBMS can choose to have multiple copies of relations on different participating sites. The benefit of data replication is increased reliability; i.e if one site fails then other sites can perform queries for other sites

3.1. Types of Distributed Database

Homogeneous: In homogeneous distributed database system, the sites [12] involved in distributed DBMS are characterized by the following:

- The same DBMS software are used at every site. i.e all DBMS can be oracle or all my SQL.
- Data are distributed across all the nodes.
- All data are managed by the distributed DBMS (so there are no exclusively local data)

Heterogeneous: Heterogeneous database systems are characterized by the following:

- Different DBMS are used at each node.
- Data are distributed across all the nodes.
- Different DBMSs may be used at each node

Though it might be easier to implement homogeneous system, heterogeneous systems are preferable because organization may have different DBMS installed at different sites and may want to access them transparently [17].

3.2. Types of Failures in Distributed Database

Several types of failures may occur in distributed database system, they are;

Transaction Failures: When a transaction fails, it aborts. Thereby the database must be restored to the state it was in before the transaction started. Transactions may [12] fail for several reasons. Some failures may be due to deadlock situation or concurrency control algorithms.

Site Failure: Site failures are usually due to software or hardware failures. These failures result in the loss of the main memory contents. In distributed database, site failures are of two types

(a) **Total Failure:** Where all the sites of a distributed system fail

(b) **Partial Failure**: Where only some of the sites of a distributed system fail.

Media Failures: Such failures refer to the failure of secondary storage devices. The failure itself may be due to head crashes, or controller failure. In these cases, [9] the media failures result in the inaccessibility of past or the entire database stored on such secondary storage.

Communication Failures: Communication failures as the name implies, are failures in the communication system between two or more sites. This will lead to network partitioning where each site, or several sites grouped together, operates independently. As such messages from one site will not reach the other sites and will therefore be lost. The reliability protocols then utilize a time out mechanism in order to detect undelivered messages [9]. A message is undelivered if the sender doesn't receive an acknowledgement. The inability of a communication network to deliver message is known as performance failure, [12].

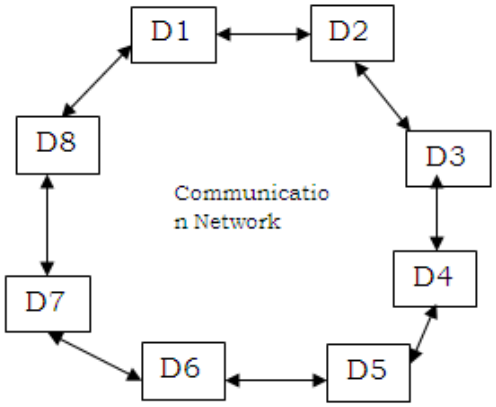


Fig 1: Distributed Database system architecture [10]

IV. Reconfiguration In DDB

System reconfiguration in a distributed database is the ability to restructure, reset or to rebuild a database in case of any failure or when it does not meet the organizational standard anymore. This is the function of communication controller or processor.

To do this, a distributed system must be able to;

- Detect failure
- Reconfigure the system so that computation may continue
- Recovery/reintegration when a site or link is repaired.

Failure Detection: Distinguishing link failure from site failure is hard, but if there is multiple link failure, then it is likely a site failure.

4.1 Steps to Reconfiguration in a DDBS

The following steps can be used to reconfigure any failed site in a Distributed database systems, these steps can be clearly seen in the activity diagram in fig. 2 below.

1. Halt all transactions that were active at a failed site

a. Making them want resources could interfere with other transactions since they may hold locks on other site (to avoid deadlock).
b. However, in case only some replicas of a data item failed, it may be possible to continue transactions that had accessed data at a failed site [1].
2. If replicated data items were at failed site, update system catalog to remove them from the list of replicas.
a. This should be reversed when failed site recovers, but additional care needs to be taken to maintain data integrity [1].
3. If a failed site was a central server for some subsystem, an election must be held to determine the new server [1].

Site Reintegration: When failed site recovers, it must catch up with all updates that it missed while it was down [1].

This can be done by;

(a) Halt all updates on system while reintegrating a site

Lock all replicas of all data items at the site, update to latest version them release locks [1].

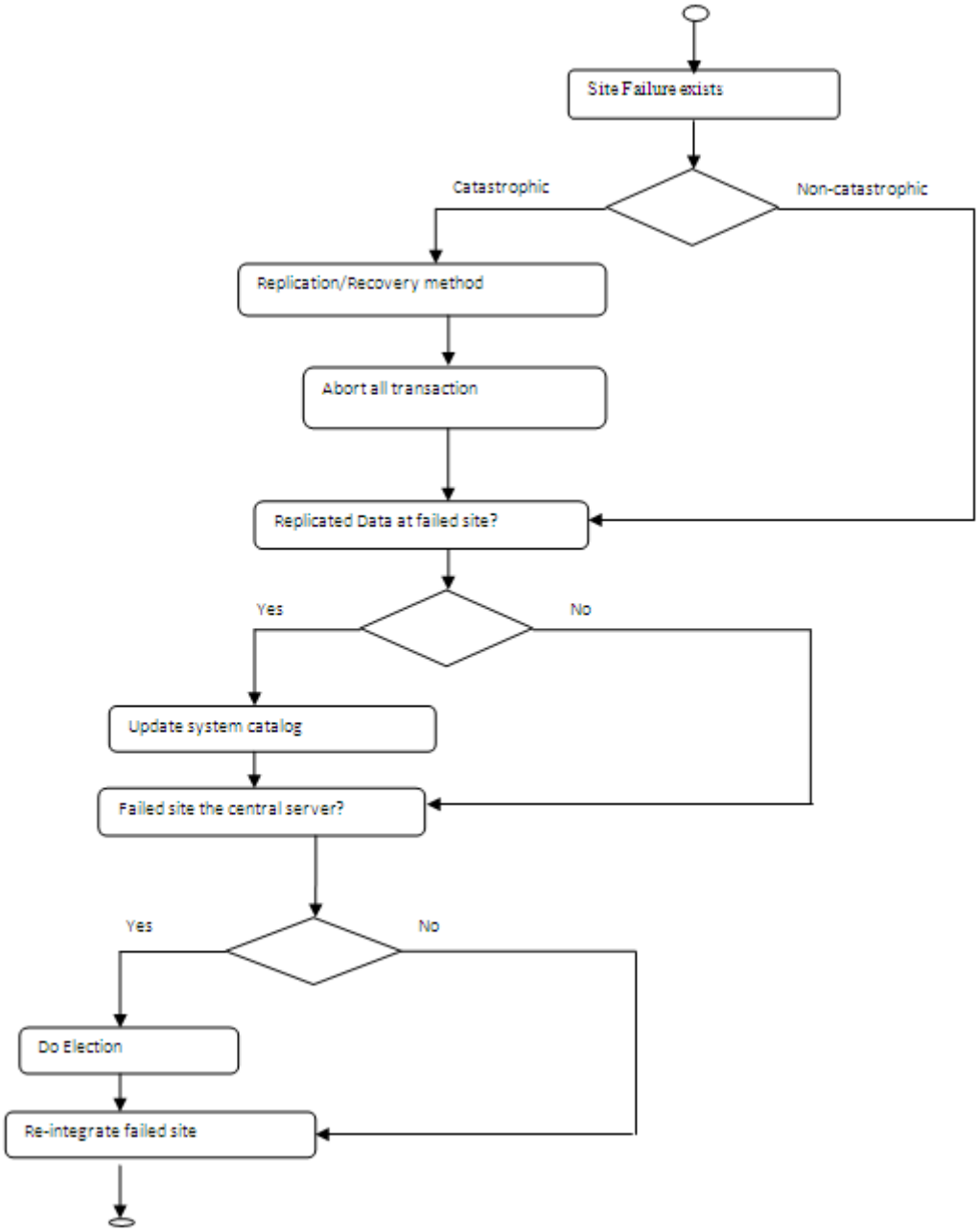
4.2Methods of System Reconfiguration in a DDBS

4.2.1Reconfiguration using recovery method

The main techniques used to handle failures like disk crash, is that of database backup. Here, the whole database and the log are periodically copied onto a cheap storage medium such as magnetic tapes. In case of a catastrophic system failure, the latest backup copy can be reloaded from the tape to the disk, and the system can be restarted. To avoid losing all the effect of transactions that have been executed since the last backup, it is customary to backup the system log by periodically copying it to magnetic tape [5].

But, if the database is not physically damaged, but has some inconsistency, the strategy is to reverse the changes that caused the inconsistency by undoing some operations. It may also be necessary to redo some operations that could have been lost during the recovery process [5].

Fragmentation: This is a process of partitioning relation of the database into several fragments and stores them in distinct sites. These fragments can then be use during reconfiguration [2].



4.2.1Reconfiguring using replicated data in the DDMS

Replication is a process whereby a database system maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.

In other words, it is the creation and maintenance of duplicate versions of database objects in a DDBS. Replication improves the performance and increase the availability of applications by providing alternate data access option. For example, user can access a local database rather than a remote server to minimize network traffic and provide location transparency, [8].

Furthermore, the application can continue to function if parts of the distributed database are down as replicas of the data might still be accessible. Database replication is needed in the case of a system failure wherein if a primary copy of the database is failed; the secondary copy will be still there to retain the data. A replication service is required to maintain system reconfiguration across these diverse environments.

Replicating data on separate computers is one strategy for ensuring that a damaged database can be quickly recovered and users can have access to data while the primary site is being restored [9]. This can be done by;

a. Halt all updates on system while reintegrating a site
b. Lock all replicas of all data items at the site, update to latest version them release locks [1].

Advantage of replication [4]

a. **Availability**: Failure of site containing relation of X-database does not result in unavailability of X if replicas exist.
b. **Parallelism**: Queries on X-database may be processed by several nodes in parallel.
c. **Reduced data transfer**: Relation of X-database is available locally at each site containing a replica of X.

Disadvantages

a. Increased Cost of updates of each replica of relation X must be updated.
b. Increased complexity of concurrency control of concurrent update to distinct replicas may lead to inconsistent data unless special concurrency control mechanism are implemented e.g. by choosing one copy as primary copy and apply concurrency control operation that primary copy [4]. Decreased performance when there are large numbers of updates.

V. Conclusion / Recommendation

Abnormalities caused by system failures in a distributed database system cannot be completely avoided. System reconfiguration remains the optional scheme over a significant range of these failures. This can be guaranteed by installing some fault tolerance scheme in the database.

In this paper, we use a replication and recovery fault-tolerant methods to reconfigure any failed site to ensure consistency, integrity and security in the distributed database, also, recovery fault-tolerant method is strongly recommended for any catastrophic failure.

References

[1]. A. Kashem, Concurrency Control in Distributed Databases, Lecture Note in Computer Science, Gazipur, India, 2005.
[2]. A. P. Sheth, Fault tolerance in a very large Distributed system: A Strawman Analysis, Unisys West Coast Research Center, 2400 Colorado Ave., Santa Monica 90406, 1989.
[3]. A. Sleit, W. AlMobaideen, S. Al-Areqi, and A. Yahya, A Dynamic Object Fragmentation and Replication Algorithm in Distributed Database Systems**,** American Journal of Applied Sciences, Vol. 4, No. 8, 2007, pp. 613 618.
[4]. A. Silberschatz, H.F. Korth, S. Sudarshan, Database System concept, 6th Edition, 2010.
[5]. B. k. Vidyasankar and T. Monoura, An optimistic Resiliency control Scheme for Distributed Database System, LNCS, Vol. 312, pp 297-307, 2005.
[6]. D. Agrawal and E. El Abbadi., On the power of Reconfiguration in Fault-tolerant Distributed Databases, Proceedings of the twenty-seventh Hawaii International conference on System science, vol.5, pp 341-350, 1994.
[7]. https://www.classle.net/book/database-backup-and-recovery-catastrophic-failures
[8]. May mar Oo. Fault tolerance by Replication of Distributed Database in P2P system using agent approach, International Journal of Computer, issue 1.vol. 4, 2010.
[9]. M. T. Ozsu, P. Valduriez, "Principles of Distributed Database Systems," (3ed), Springer, New York, 2011, pp. 459-495
[10]. N. Eva Wu, Matthew C. Ruschmann, and Mark H. Linderman, Fault-Tolerant control of a distributed database system, Journal of control science and Engineering, Article 310652, volume 2, pg 13, 2008.
[11]. N. E. Wu., Reconfiguration of C2 architecture for improved availability to support air operations, IEEE Transaction on Aerospace and electronics, Vol. 43, no 2, pp. 795-805, 2007.
[12]. N. Kumar, S. Bilgaiyan , S. Sagnika, An Overview of Transparency in Homogeneous Distributed Database System, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 10, October 2013
[13]. O'Brien, J. & Marakas, G.M. (2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin
[14]. Sang Hyuk Son, Replicated Data management in Distributed Database system, ACM Signed, Vol.17, Issue 4, New York, USA, pp 62-69, 1988.
[15]. T. Ananthapadmanabha, R. Prakash, M. K. Pujar, A.Gangadhara and M. Gangadhara, System reconfiguration and service restoration of primary distribution system augmented by capacitors using a new level-wise approach, Journal of Electrical and Electronic Engineering research vol. 3 (3), pp.42-51, 2011
[16]. T. Connlly and C. Begg, Database solution: A step by step Guide to Building Databases, pearson /Addison Wesley, New York, Ny, USA, 2nd edition, 2004.
[17]. W. Cellary, E. Gelenbe, and T. Morzy. Concurrency Control in Distributed Database System, North- Holland, 1988.

Authors' Brief

Silas Abasiama Ita holds a B.Sc (Hons) in Computer Science/Mathematics at the University of Port-Harcourt, Rivers State, Nigeria, in 2009. She is currently concluding her M.Sc Degree programme in Computer Science at University of Port Harcourt, Nigeria. Her research interest includes: Distributed Database/Distributed Processing, Network/Data security, Modeling, Software Engineering and Artificial Intelligence. She is an Associate member of CISCO, member of IEEE and IEEE-Computer Society. She can be contacted via abasiama_silas@uniport.edu.ng and +2348034765201

Prince Oghenekaro Asagba had his B.Sc. degree in Computer Science at the University of Nigeria, Nsukka, in 1991, M.Sc. degree in Computer Science at the University of Benin in April, 1998, and a Ph.D degree in Computer Science at the University of Port Harcourt in March, 2009. He is a Senior Lecturer and a visiting lecturer to several Universities in Nigeria since 2010. His research interest includes: Network Security, Information Security, Network Analysis, Modeling, Database Management Systems, Object-oriented Design, and Programming. He has published several articles in Learned Journals both in Nigeria and Internationally. He has authored and coauthored several books in Computer Science. He is a member of Nigeria Computer Society (NCS) and Computer Professional Registration Council of Nigeria (CPN).

Igiri Chinwe Peace holds a B.Eng. in Computer Science and Engineering from Enugu State University of Science & Technology, in 2001. She is a currently at the concluding stages of Master's degree programme at the department of Computer Science, University of Port Harcourt, Nigeria. Her research areas are data base technology, data mining and software engineering. She is a member of Nigeria Society of Engineering, ACM and IEEE-Computer Society. She can be reached via chynkemdirim@gmail.com and +2348036012095.